First-Order Logic
Syntax
Common Sense Reasoning
Example, adapted from Lenat

You are told: John drove to the grocery store and bought a pound of noodles, a pound of ground beef, and two pounds of tomatoes.

• Is John 3 years old?
• Is John a child?
• What will John do with the purchases?
• Did John have any money?
• Does John have less money after going to the store?
• Did John buy at least two tomatoes?
• Were the tomatoes made in the supermarket?
• Did John buy any meat?
• Is John a vegetarian?
• Will the tomatoes fit in John’s car?

• Can Propositional Logic support these inferences?
Outline for First-Order Logic (FOL, also called FOPC)

• Propositional Logic is **Useful** --- but has **Limited Expressive Power**

• First Order Predicate Calculus (FOPC), or First Order Logic (FOL).
  – FOPC has greatly expanded expressive power, though still limited.

• New Ontology
  – The world consists of **OBJECTS** (for propositional logic, the world was facts).
  – **OBJECTS** have **PROPERTIES** and engage in **RELATIONS** and **FUNCTIONS**.

• New Syntax
  – **Constants**, **Predicates**, **Functions**, **Properties**, **Quantifiers**.

• New Semantics
  – Meaning of new syntax.

• Knowledge engineering in FOL

• Unification Inference in FOL
FOL Syntax: You will be expected to know

- **FOPC syntax**
  - Syntax: Sentences, predicate symbols, function symbols, constant symbols, variables, quantifiers
- **De Morgan’s rules for quantifiers**
  - connections between ∀ and ∃
- **Nested quantifiers**
  - Difference between “∀ x ∃ y P(x, y)” and “∃ x ∀ y P(x, y)”
  - ∀ x ∃ y Likes(x, y) --- “Everybody likes somebody.”
  - ∃ x ∀ y Likes(x, y) --- “Somebody likes everybody.”
- **Translate simple English sentences to FOPC and back**
  - ∀ x ∃ y Likes(x, y) ↔ “Everyone has someone that they like.”
  - ∃ x ∀ y Likes(x, y) ↔ “There is someone who likes every person.”
Pros and cons of propositional logic

😊 Propositional logic is **declarative**
- Knowledge and inference are separate

😊 Propositional logic allows **partial/disjunctive/negated information**
- unlike most programming languages and databases

😊 Propositional logic is **compositional**:
- meaning of $B_{1,1} \land P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$

😊 Meaning in propositional logic is **context-independent**
- unlike natural language, where meaning depends on context

SORROW Propositional logic has **limited expressive power**
- E.g., cannot say “Pits cause breezes in adjacent squares.”
  - except by writing one sentence for each square
- Needs to refer to objects in the world,
- Needs to express general rules
First-Order Logic (FOL), also called First-Order Predicate Calculus (FOPC)

- Propositional logic assumes the world contains facts.

- First-order logic (like natural language) assumes the world contains
  - **Objects**: people, houses, numbers, colors, baseball games, wars, ...
  - **Functions**: father of, best friend, one more than, plus, ...
    - Function arguments are objects; function returns an object
  - **Objects generally correspond to English NOUNS**

- **Predicates/Relations/Properties**: red, round, prime, brother of, bigger than, part of, comes between, ...
  - Predicate arguments are objects; predicate returns a truth value
  - **Predicates generally correspond to English VERBS**
    - **First argument is generally the subject, the second the object**
    - Hit(Bill, Ball) usually means “Bill hit the ball.”
    - Likes(Bill, IceCream) usually means “Bill likes IceCream.”
    - Verb(Noun1, Noun2) usually means “Noun1 verb noun2.”
Aside: First-Order Logic (FOL) vs. Second-Order Logic

- First Order Logic (FOL) allows variables and general rules
  - “First order” because quantified variables represent objects.
  - “Predicate Calculus” because it quantifies over predicates on objects.
    - E.g., “Integral Calculus” quantifies over functions on numbers.

- Aside: Second Order logic
  - “Second order” because quantified variables can also represent predicates and functions.
    - E.g., can define “Transitive Relation,” which is beyond FOPC.

- Aside: In FOL we can state that a relationship is transitive
  - E.g., BrotherOf is a transitive relationship
    - $\forall x, y, z \text{ BrotherOf}(x,y) \land \text{ BrotherOf}(y,z) \Rightarrow \text{ BrotherOf}(x,z)$

- Aside: In Second Order logic we can define “Transitive”
  - $\forall P, x, y, z \text{ Transitive}(P) \Leftrightarrow ( P(x,y) \land P(y,z) \Rightarrow P(x,z) )$
  - Then we can state directly, Transitive(BrotherOf)
FOL (or FOPC) Ontology:
What kind of things exist in the world?
What do we need to describe and reason about?
Objects --- with their relations, functions, predicates, properties, and general rules.
## Syntax of FOL: Basic elements

<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constants</td>
<td>KingJohn, 2, UCI,...</td>
</tr>
<tr>
<td>Predicates</td>
<td>Brother, &gt;,...</td>
</tr>
<tr>
<td>Functions</td>
<td>Sqrt, LeftLegOf,...</td>
</tr>
<tr>
<td>Variables</td>
<td>x, y, a, b,...</td>
</tr>
<tr>
<td>Quantifiers</td>
<td>∀, ∃</td>
</tr>
<tr>
<td>Connectives</td>
<td>¬, ∧, ∨, ⇒, ⇔ (standard)</td>
</tr>
<tr>
<td>Equality</td>
<td>= (but causes difficulties....)</td>
</tr>
</tbody>
</table>
Syntax of FOL: Basic syntax elements are symbols

- **Constant** Symbols (correspond to English nouns)
  - Stand for objects in the world.
    - E.g., KingJohn, 2, UCI, ...

- **Predicate** Symbols (correspond to English verbs)
  - Stand for relations (maps a tuple of objects to a **truth-value**)
    - E.g., Brother(Richard, John), greater_than(3,2), ...
    - P(x, y) is usually read as “x is P of y.”
      - E.g., Mother(Ann, Sue) is usually “Ann is Mother of Sue.”

- **Function** Symbols (correspond to English nouns)
  - Stand for functions (maps a tuple of objects to an **object**)
    - E.g., Sqrt(3), LeftLegOf(John), ...

- **Model** (world) = set of domain objects, relations, functions
- **Interpretation** maps symbols onto the model (world)
  - Very many interpretations are possible for each KB and world!
  - Job of the KB is to rule out models inconsistent with our knowledge.
Syntax: Relations, Predicates, Properties, Functions

- Mathematically, all the Relations, Predicates, Properties, and Functions CAN BE represented simply as sets of $m$-tuples of objects:

- Let $W$ be the set of objects in the world.

- Let $W^m = W \times W \times \ldots \ (m \text{ times}) \ldots \times W$
  - The set of all possible $m$-tuples of objects from the world

- An $m$-ary Relation is a subset of $W^m$.
  - Example: Let $W = \{John, Sue, Bill\}$
  - Then $W^2 = \{<John, John>, <John, Sue>, \ldots, <Sue, Sue>\}$
  - E.g., MarriedTo = $\{<John, Sue>, <Sue, John>\}$
  - E.g., FatherOf = $\{<John, Bill>\}$

- Analogous to a constraint in CSPs
  - The constraint lists the $m$-tuples that satisfy it.
  - The relation lists the $m$-tuples that participate in it.
Syntax: Relations, Predicates, Properties, Functions

- **A Predicate** is a list of \(m\)-tuples making the predicate true.
  - E.g., \(\text{PrimeFactorOf} = \{<2,4>, <2,6>, <3,6>, <2,8>, <3,9>, \ldots\}\)
  - This is the same as an \(m\)-ary Relation.
  - Predicates (and properties) generally correspond to English verbs.

- **A Property** lists the \(m\)-tuples that have the property.
  - Formally, it is a predicate that is true of tuples having that property.
  - E.g., \(\text{IsRed} = \{<\text{Ball-5}>, <\text{Toy-7}>, <\text{Car-11}>, \ldots\}\)
  - This is the same as an \(m\)-ary Relation.

- **A Function** CAN BE represented as an \(m\)-ary relation
  - the first \((m-1)\) objects are the arguments and the \(m^{th}\) is the value.
  - E.g., \(\text{Square} = \{<1, 1>, <2, 4>, <3, 9>, <4, 16>, \ldots\}\)

- **An Object** CAN BE represented as a function of zero arguments that returns the object.
  - This is just a 1-ary relationship.
Syntax of FOL: Terms

- **Term** = logical expression that **refers to an object**

- **There are two kinds of terms:**
  - **Constant Symbols** stand for (or name) objects:
    - E.g., KingJohn, 2, UCI, Wumpus, ...
  
  - **Function Symbols** map tuples of objects to an object:
    - E.g., LeftLeg(KingJohn), Mother(Mary), Sqrt(x)
    - This is nothing but a complicated kind of name
      - No “subroutine” call, no “return value”
Syntax of FOL: Atomic Sentences

• **Atomic Sentences** state facts (logical truth values).
  – An **atomic sentence** is a Predicate symbol, optionally followed by a parenthesized list of any argument terms
  – E.g., *Married( Father(Richard), Mother(John) )*  
  – An **atomic sentence** asserts that some relationship (some predicate) holds among the objects that are its arguments.

• An **Atomic Sentence is true** in a given model if the relation referred to by the predicate symbol holds among the objects (terms) referred to by the arguments.
Syntax of FOL: Atomic Sentences

• Atomic sentences in logic state facts that are true or false.

• Properties and \( m \)-ary relations do just that:
  - LargerThan(2, 3) is false.
  - BrotherOf(Mary, Pete) is false.
  - Married(Father(Richard), Mother(John)) could be true or false.

Properties and \( m \)-ary relations are Predicates that are true or false.

• Note: Functions refer to objects, do not state facts, and form no sentence:
  - Brother(Pete) refers to John (his brother) and is neither true nor false.
  - Plus(2, 3) refers to the number 5 and is neither true nor false.

• BrotherOf( Pete, Brother(Pete) ) is True.

  Binary relation is a truth value. Function refers to John, an object in the world, i.e., John is Pete's brother.
  (Works well iff John is Pete's only brother.)
Syntax of FOL: Connectives & Complex Sentences

• **Complex Sentences** are formed in the same way, and are formed using the same logical connectives, as we already know from propositional logic.

• **The Logical Connectives:**
  - $\iff$ biconditional
  - $\implies$ implication
  - $\land$ and
  - $\lor$ or
  - $\neg$ negation

• **Semantics** for these logical connectives are the same as we already know from propositional logic.
We make complex sentences with connectives (just like in propositional logic).

$$\neg \text{Brother}(\text{LeftLeg}(\text{Richard}), \text{John}) \lor (\text{Democrat}(\text{Bush}))$$
Examples

- Brother(Richard, John) ∧ Brother(John, Richard)
- King(Richard) ∨ King(John)
- King(John) => ¬ King(Richard)
- LessThan(Plus(1,2),4) ∧ GreaterThan(1,2)

(Semantics of complex sentences are the same as in propositional logic)
Syntax of FOL: Variables

- **Variables** range over objects in the world.

- A **variable** is like a **term** because it represents an object.

- A **variable** may be used wherever a **term** may be used.
  - **Variables** may be arguments to functions and predicates.

- (A **term with NO variables** is called a **ground term**.)
- (A **variable not bound by a quantifier** is called **free**.)
Syntax of FOL: Logical Quantifiers

• There are two **Logical Quantifiers**:
  – **Universal**: \( \forall x P(x) \) means “For all x, \( P(x) \).”
    • The “upside-down A” reminds you of “ALL.”
  – **Existential**: \( \exists x P(x) \) means “There exists x such that, \( P(x) \).”
    • The “backward E” reminds you of “EXISTS.”

• Syntactic “sugar” --- we really only need one quantifier.
  – \( \forall x P(x) \equiv \neg \exists x \neg P(x) \)
  – \( \exists x P(x) \equiv \neg \forall x \neg P(x) \)
  – You can ALWAYS convert one quantifier to the other.

• **RULES**: \( \forall \equiv \neg \exists \) and \( \exists \equiv \neg \forall \)

• **RULE**: To move negation “in” across a quantifier,
  change the quantifier to “the other quantifier”
  and negate the predicate on “the other side.”
  – \( \neg \forall x P(x) \equiv \exists x \neg P(x) \)
  – \( \neg \exists x P(x) \equiv \forall x \neg P(x) \)
Universal Quantification ∀

- ∀ means “for all”
- Allows us to make statements about all objects that have certain properties
- Can now state general rules:
  
  ∀ x King(x) => Person(x) “All kings are persons.”
  ∀ x Person(x) => HasHead(x) “Every person has a head.”
  ∀ i Integer(i) => Integer(plus(i,1)) “If i is an integer then i+1 is an integer.”

Note that
 ∀ x King(x) ∧ Person(x) is not correct!
This would imply that all objects x are Kings and are People

∀ x King(x) => Person(x) is the correct way to say this

Note that => is the natural connective to use with ∀ .
Universal Quantification ∀

- Universal quantification is equivalent to:
  - Conjunction of all sentences obtained by substitution of an object for the quantified variable.

- All Cats are Mammals.
  - ∀x Cat(x) ⇒ Mammal(x)

- Conjunction of all sentences obtained by substitution of an object for the quantified variable:
  Cat(Spot) ⇒ Mammal(Spot) ∧
  Cat(Rick) ⇒ Mammal(Rick) ∧
  Cat(LAX) ⇒ Mammal(LAX) ∧
  Cat(Shayama) ⇒ Mammal(Shayama) ∧
  Cat(France) ⇒ Mammal(France) ∧
  Cat(Felix) ⇒ Mammal(Felix) ∧
  ...

Existential Quantification $\exists$

- $\exists x$ means "there exists an $x$ such that..." (at least one object $x$)
- Allows us to make statements about some object without naming it
- Examples:

  $\exists x \text{ King}(x)$ "Some object is a king."
  
  $\exists x \text{ Lives\_in}(\text{John}, \text{Castle}(x))$ "John lives in somebody’s castle."
  
  $\exists i \text{ Integer}(i) \land \text{GreaterThan}(i,0)$ "Some integer is greater than zero."

Note that $\land$ is the natural connective to use with $\exists$

(And note that $\Rightarrow$ is the natural connective to use with $\forall$)
Existential Quantification $\exists$

- Existential quantification is equivalent to:
  - Disjunction of all sentences obtained by substitution of an object for the quantified variable.

- Spot has a sister who is a cat.
  - $\exists x \text{ Sister}(x, \text{Spot}) \land \text{Cat}(x)$

- Disjunction of all sentences obtained by substitution of an object for the quantified variable:

  $\text{Sister}(\text{Spot}, \text{Spot}) \land \text{Cat}(\text{Spot}) \lor$

  $\text{Sister}(\text{Rick}, \text{Spot}) \land \text{Cat}(\text{Rick}) \lor$

  $\text{Sister}(\text{LAX}, \text{Spot}) \land \text{Cat}(\text{LAX}) \lor$

  $\text{Sister}(\text{Shayama}, \text{Spot}) \land \text{Cat}(\text{Shayama}) \lor$

  $\text{Sister}(\text{France}, \text{Spot}) \land \text{Cat}(\text{France}) \lor$

  $\text{Sister}(\text{Felix}, \text{Spot}) \land \text{Cat}(\text{Felix}) \lor$

  ...
Combining Quantifiers --- Order (Scope)

The order of “unlike” quantifiers is important.

Like nested variable scopes in a programming language
Like nested ANDs and ORs in a logical sentence

∀ x  ∃ y  Loves(x,y)
  – For everyone (“all x”) there is someone (“exists y”) whom they love.
  – There might be a different y for each x (y is inside the scope of x)

∃ y  ∀ x  Loves(x,y)
  – There is someone (“exists y”) whom everyone loves (“all x”).
  – Every x loves the same y (x is inside the scope of y)

Clearer with parentheses:  ∃ y ( ∀ x  Loves(x,y) )

The order of “like” quantifiers does not matter.

Like nested ANDs and ANDs in a logical sentence

∀x  ∀y  P(x, y) ≡ ∀y  ∀x  P(x, y)
∃x  ∃y  P(x, y) ≡ ∃y  ∃x  P(x, y)
Connections between Quantifiers

- Asserting that all $x$ have property $P$ is the same as asserting that does not exist any $x$ that does not have the property $P$

$$∀x \text{ Likes}(x, \text{CS-171 class}) \iff \neg ∃x \neg \text{Likes}(x, \text{CS-171 class})$$

- Asserting that there exists an $x$ with property $P$ is the same as asserting that not all $x$ do not have the property $P$

$$∃x \text{ Likes}(x, \text{IceCream}) \iff \neg ∀x \neg \text{Likes}(x, \text{IceCream})$$

In effect:
- $∀$ is a conjunction over the universe of objects
- $∃$ is a disjunction over the universe of objects
  Thus, DeMorgan’s rules can be applied
## De Morgan’s Law for Quantifiers

<table>
<thead>
<tr>
<th>De Morgan’s Rule</th>
<th>Generalized De Morgan’s Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \land Q \equiv \neg (\neg P \lor \neg Q)$</td>
<td>$\forall x \ P \equiv \neg \exists x (\neg P)$</td>
</tr>
<tr>
<td>$P \lor Q \equiv \neg (\neg P \land \neg Q)$</td>
<td>$\exists x \ P \equiv \neg \forall x (\neg P)$</td>
</tr>
<tr>
<td>$\neg (P \land Q) \equiv \neg P \lor \neg Q$</td>
<td>$\neg \forall x \ P \equiv \exists x (\neg P)$</td>
</tr>
<tr>
<td>$\neg (P \lor Q) \equiv \neg P \land \neg Q$</td>
<td>$\neg \exists x \ P \equiv \forall x (\neg P)$</td>
</tr>
</tbody>
</table>

Rule is simple: if you bring a negation inside a disjunction or a conjunction, always switch between them (or → and, and → or).
Aside: More syntactic sugar --- uniqueness

• $\exists! \ x$ is “syntactic sugar” for “There exists a unique $x$”
  - “There exists one and only one $x$”
  - “There exists exactly one $x$”
  - Sometimes $\exists!$ is written as $\exists^1$

• For example, $\exists! \ x \text{PresidentOfTheUSA}(x)$
  - “There is exactly one PresidentOfTheUSA.”

• This is just syntactic sugar:
  - $\exists! \ x \ P(x)$ is the same as $\exists \ x \ P(x) \land (\forall \ y \ P(y) \Rightarrow (x = y))$
Equality

- $term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object

- E.g., definition of Sibling in terms of Parent:

\[
\forall x,y \ Sibling(x,y) \iff \\
\quad \neg (x = y) \land \\
\quad \exists m,f \ (m = f) \land \text{Parent}(m,x) \land \text{Parent}(f,x) \\
\quad \land \text{Parent}(m,y) \land \text{Parent}(f,y)
\]

Equality can make reasoning much more difficult!

(See R&N, section 9.5.5, page 353)

You may not know when two objects are equal.

E.g., Ancients did not know (MorningStar = EveningStar = Venus)

You may have to prove $x = y$ before proceeding

E.g., a resolution prover may not know 2+1 is the same as 1+2
Syntactic Ambiguity

- FOPC provides many ways to represent the same thing.
- E.g., “Ball-5 is red.”
  - HasColor(Ball-5, Red)
    - Ball-5 and Red are objects related by HasColor.
  - Red(Ball-5)
    - Red is a unary predicate applied to the Ball-5 object.
  - HasProperty(Ball-5, Color, Red)
    - Ball-5, Color, and Red are objects related by HasProperty.
  - ColorOf(Ball-5) = Red
    - Ball-5 and Red are objects, and ColorOf() is a function.
  - HasColor(Ball-5(), Red())
    - Ball-5() and Red() are functions of zero arguments that both return an object, which objects are related by HasColor.
  - ...

- This can GREATLY confuse a pattern-matching reasoner.
  - Especially if multiple people collaborate to build the KB, and they all have different representational conventions.
Syntactic Ambiguity --- Partial Solution

• FOL can be TOO expressive, can offer TOO MANY choices

• Likely confusion, especially for teams of Knowledge Engineers

• Different team members can make different representation choices
  – E.g., represent “Ball43 is Red.” as:
    • a predicate (= verb)? E.g., “Red(Ball43)”?
    • an object (= noun)? E.g., “Red = Color(Ball43))”?
    • a property (= adjective)? E.g., “HasProperty(Ball43, Red)”?

• PARTIAL SOLUTION:
  – An upon-agreed ontology that settles these questions
  – Ontology = what exists in the world & how it is represented
  – The Knowledge Engineering teams agrees upon an ontology BEFORE they begin encoding knowledge
Fun with sentences

Brothers are siblings
Brothers are siblings

\[ \forall x, y \ Brothe r(x, y) \Rightarrow Sibling(x, y). \]

“Sibling” is symmetric
Fun with sentences

Brothers are siblings

\[ \forall x, y \; \text{Brother}(x, y) \implies \text{Sibling}(x, y). \]

“Sibling” is symmetric

\[ \forall x, y \; \text{Sibling}(x, y) \iff \text{Sibling}(y, x). \]

One’s mother is one’s female parent
Fun with sentences

Brothers are siblings

\[ \forall x, y \; \text{Brother}(x, y) \implies \text{Sibling}(x, y). \]

“Sibling” is symmetric

\[ \forall x, y \; \text{Sibling}(x, y) \iff \text{Sibling}(y, x). \]

One’s mother is one’s female parent

\[ \forall x, y \; \text{Mother}(x, y) \iff (\text{Female}(x) \land \text{Parent}(x, y)). \]

A first cousin is a child of a parent’s sibling
Fun with sentences

Brothers are siblings

\[ \forall x, y \; \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y). \]

“Sibling” is symmetric

\[ \forall x, y \; \text{Sibling}(x, y) \iff \text{Sibling}(y, x). \]

One’s mother is one’s female parent

\[ \forall x, y \; \text{Mother}(x, y) \iff (\text{Female}(x) \land \text{Parent}(x, y)). \]

A first cousin is a child of a parent’s sibling

\[ \forall x, y \; \text{FirstCousin}(x, y) \iff \exists p, ps \; \text{Parent}(p, x) \land \text{Sibling}(ps, p) \land \text{Parent}(ps, y) \]
More fun with sentences

- “All persons are mortal.”
- [Use: Person(x), Mortal (x) ]
More fun with sentences

- “All persons are mortal.”
- \[\forall x \text{ Person}(x) \Rightarrow \text{Mortal}(x)\]

- \[\forall x \neg\text{Person}(x) \lor \text{Mortal}(x)\]

**Common Mistakes:**

- \[\forall x \text{ Person}(x) \land \text{Mortal}(x)\]
More fun with sentences

• “Fifi has a sister who is a cat.”
• \[ \text{Use: Sister(Fifi, } x), \text{ Cat}(x) \]
More fun with sentences

• “Fifi has a sister who is a cat.”
• [Use: Sister(Fifi, x), Cat(x) ]
• ⋄
• \( \exists x \text{ Sister}(Fifi, x) \land \text{Cat}(x) \)

• **Common Mistakes:**
• \( \exists x \text{ Sister}(Fifi, x) \Rightarrow \text{Cat}(x) \)
More fun with sentences

- “For every food, there is a person who eats that food.”
- [Use: Food(x), Person(y), Eats(y, x) ]
More fun with sentences

• “For every food, there is a person who eats that food.”
• [Use: Food(x), Person(y), Eats(y, x) ]

• \( \forall x \exists y \text{ Food}(x) \rightarrow [ \text{ Person}(y) \land \text{ Eats}(y, x) ] \)
• \( \forall x \text{ Food}(x) \rightarrow \exists y [ \text{ Person}(y) \land \text{ Eats}(y, x) ] \)
• \( \forall x \exists y \neg \text{ Food}(x) \lor [ \text{ Person}(y) \land \text{ Eats}(y, x) ] \)
• \( \forall x \exists y [ \neg \text{ Food}(x) \lor \text{ Person}(y) ] \land [ \neg \text{ Food}(x) \lor \text{ Eats}(y, x) ] \)
• \( \forall x \exists y [ \text{ Food}(x) \Rightarrow \text{ Person}(y) ] \land [ \text{ Food}(x) \Rightarrow \text{ Eats}(y, x) ] \)

• **Common Mistakes:**
• \( \forall x \exists y [ \text{ Food}(x) \land \text{ Person}(y) ] \Rightarrow \text{ Eats}(y, x) \)
• \( \forall x \exists y \text{ Food}(x) \land \text{ Person}(y) \land \text{ Eats}(y, x) \)
More fun with sentences

- “Every person eats every food.”
- [Use: Person (x), Food (y), Eats(x, y) ]
More fun with sentences

• “Every person eats every food.”
• [Use: Person (x), Food (y), Eats(x, y) ]
• $\forall x \forall y [ \text{Person}(x) \land \text{Food}(y) ] \Rightarrow \text{Eats}(x, y)$
• $\forall x \forall y \neg \text{Person}(x) \lor \neg \text{Food}(y) \lor \text{Eats}(x, y)$
• $\forall x \forall y \text{Person}(x) \Rightarrow [ \text{Food}(y) \Rightarrow \text{Eats}(x, y) ]$
• $\forall x \forall y \text{Person}(x) \Rightarrow [ \neg \text{Food}(y) \lor \text{Eats}(x, y) ]$
• $\forall x \forall y \neg \text{Person}(x) \lor [ \text{Food}(y) \Rightarrow \text{Eats}(x, y) ]$
• **Common Mistakes:**
• $\forall x \forall y \text{Person}(x) \Rightarrow [ \text{Food}(y) \land \text{Eats}(x, y) ]$
• $\forall x \forall y \text{Person}(x) \land \text{Food}(y) \land \text{Eats}(x, y)$
More fun with sentences

- “All greedy kings are evil.”
- [Use: King(x), Greedy(x), Evil(x) ]
More fun with sentences

• “All greedy kings are evil.”
• \[ \forall x \ [ \text{Greedy}(x) \land \text{King}(x) ] \Rightarrow \text{Evil}(x) \]
• \[ \forall x \ [ \neg \text{Greedy}(x) \lor \neg \text{King}(x) \lor \text{Evil}(x) ] \]
• \[ \forall x \ [ \text{Greedy}(x) \Rightarrow [ \text{King}(x) \Rightarrow \text{Evil}(x) ] \]

Common Mistakes:
• \[ \forall x \ [ \text{Greedy}(x) \land \text{King}(x) \land \text{Evil}(x) ] \]
More fun with sentences

- “Everyone has a favorite food.”
- [Use: Person(x), Food(y), Favorite(y, x) ]
More fun with sentences

• “Everyone has a favorite food.”
  [Use: Person(x), Food(y), Favorite(y, x) ]
• \( \forall x \exists y \text{ Person}(x) \implies [ \text{Food}(y) \land \text{Favorite}(y, x) ] \)
• \( \forall x \text{ Person}(x) \implies \exists y [ \text{Food}(y) \land \text{Favorite}(y, x) ] \)
• \( \forall x \exists y \neg \text{Person}(x) \lor [ \text{Food}(y) \land \text{Favorite}(y, x) ] \)
• \( \forall x \exists y [ \neg \text{Person}(x) \lor \text{Food}(y) ] \land [ \neg \text{Person}(x) \lor \text{Favorite}(y, x) ] \)
• \( \forall x \exists y [ \text{Person}(x) \implies \text{Food}(y) ] \land [ \text{Person}(x) \implies \text{Favorite}(y, x) ] \)

• **Common Mistakes:**
  • \( \forall x \exists y [ \text{Person}(x) \land \text{Food}(y) ] \implies \text{Favorite}(y, x) \)
  • \( \forall x \exists y \text{ Person}(x) \land \text{Food}(y) \land \text{Favorite}(y, x) \)
More fun with sentences

• “There is someone at UCI who is smart.”
  [Use: Person(x), At(x, UCI), Smart(x) ]
•
More fun with sentences

- “There is someone at UCI who is smart.”
  - [Use: Person(x), At(x, UCI), Smart(x) ]
-
- $\exists x \text{ Person}(x) \land \text{At}(x, \text{UCI}) \land \text{Smart}(x)$

**Common Mistakes:**
- $\exists x [ \text{Person}(x) \land \text{At}(x, \text{UCI}) ] \Rightarrow \text{Smart}(x)$
More fun with sentences

• “Everyone at UCI is smart.”
• [Use: Person(x), At(x, UCI), Smart(x) ]
More fun with sentences

• “Everyone at UCI is smart.”
  [Use: Person(x), At(x, UCI), Smart(x)]
  
  • $\forall x \ [\text{Person}(x) \land \text{At}(x, \text{UCI})] \implies \text{Smart}(x)$
  • $\forall x \ \neg[\text{Person}(x) \land \text{At}(x, \text{UCI})] \lor \text{Smart}(x)$
  • $\forall x \ \neg\text{Person}(x) \lor \neg\text{At}(x, \text{UCI}) \lor \text{Smart}(x)$

• **Common Mistakes:**
  • $\forall x \ \text{Person}(x) \land \text{At}(x, \text{UCI}) \land \text{Smart}(x)$
  • $\forall x \ \text{Person}(x) \implies [\text{At}(x, \text{UCI}) \land \text{Smart}(x)]$
More fun with sentences

• “Every person eats some food.”
• [Use: Person (x), Food (y), Eats(x, y) ]
More fun with sentences

• “Every person eats some food.”
  [Use: Person (x), Food (y), Eats(x, y) ]

• \( \forall x \exists y \, \text{Person}(x) \Rightarrow [ \text{Food}(y) \land \text{Eats}(x, y) ] \)
• \( \forall x \, \text{Person}(x) \Rightarrow \exists y [ \text{Food}(y) \land \text{Eats}(x, y) ] \)
• \( \forall x \, \exists y \, \neg \text{Person}(x) \lor [ \text{Food}(y) \land \text{Eats}(x, y) ] \)
• \( \forall x \, \exists y [ \neg \text{Person}(x) \lor \text{Food}(y) ] \land [ \neg \text{Person}(x) \lor \text{Eats}(x, y) ] \)

• **Common Mistakes:**
• \( \forall x \, \exists y [ \text{Person}(x) \land \text{Food}(y) ] \Rightarrow \text{Eats}(x, y) \)
• \( \forall x \, \exists y \, \text{Person}(x) \land \text{Food}(y) \land \text{Eats}(x, y) \)

•
More fun with sentences

- “Some person eats some food.”
  - [Use: Person (x), Food (y), Eats(x, y)]
More fun with sentences

- “Some person eats some food.”
  [Use: Person (x), Food (y), Eats(x, y) ]
- \( \exists x \exists y \text{ Person}(x) \land \text{Food}(y) \land \text{Eats}(x, y) \)
- **Common Mistakes:**
  - \( \exists x \exists y \left[ \text{Person}(x) \land \text{Food}(y) \right] \Rightarrow \text{Eats}(x, y) \)
Summary

• First-order logic:
  – Much more expressive than propositional logic
  – Allows objects and relations as semantic primitives
  – Universal and existential quantifiers

• Syntax: constants, functions, predicates, equality, quantifiers

• Nested quantifiers
  – Order of unlike quantifiers matters (the outer scopes the inner)
    • Like nested ANDs and ORs
  – Order of like quantifiers does not matter
    • like nested ANDS and ANDs

• Translate simple English sentences to FOPC and back